Volume

4

BIO-TECH MEDICAL SOFTWARE, INC.

BioTrackTHC JSON API

Washington State
Liquor Control Board

BioTrackTHC

BioTrackTHC API

For questions regarding this API, please call 1-800-779-4094 or email waquestions@biotrackthc.com

BIO-TECH MEDICAL SOFTWARE, INC.

BioTrackTHC JSON API

Fort Lauderdale, FL
Phone 800.797.4711

Table of Contents

Prefix: About This Document

Welcome to BioTrackTHC JSON platform. This manual serves as a comprehensive guide that details the various functions and data points that are relevant for the BioTrackTHC traceability system. This document is being released to the public in draft form ahead of schedule to expedite the integration process for commercial entities that intend to serve the producer, processor and retail establishments within the state of Washington.

Please note: There WILL be changes to this document. This may include pairing down of existing structures or additions to the specification based on legal requirements.

Although this document is public and may be read by anyone; much of it assumes that the reader has a basic understanding of web technologies and programming interfaces. It is geared towards individuals looking to interface directly to the state traceability system without utilizing the official state web interface. The official state web interface will be available at no cost for individuals who wish to upload their data without a commercial application. However, the official web interface is intended to only collect the minimum amount of information for the state compliance and does not collect information related to e.g. sales; every licensee is responsible for keeping their own business records.

BMSI and WSLCB shall in no event be liable to any party for direct, indirect, special, general, incidental, or consequential damages arising from the use of its documentation, or any derivative works thereof, even if BMSI or WSLCB have been advised of the possibility of such damage. The documentation, and any derivative works are provided on an as-is basis, and thus comes with absolutely no warranty, either express or implied. This disclaimer includes, but is not limited to, implied warranties of merchantability, fitness for any particular purpose, and non-infringement. BMSI and WSLCB have no obligation to provide maintenance, support, or updates.

Information in this document is subject to change without notice and should not be construed as a commitment by BMSI or WSLCB. While the information contained herein is believed to be accurate, BMSI and WSLCB assume no responsibility for any errors and/or omissions that may appear in this document.

That being said, we look forward to working with the industry to finalize and solidify the world's first official marijuana traceability API.

Changes

Before diving in, there have been a number of changes since the initial draft. The current draft includes Washington specific language and functions. The inventory typing system has been greatly expanded to cover all of the various types of inventory that have been defined with limits as delineated in law and rules.

Inventory Types

5	Kief
6	Flower
7	Clone
9	Other Plant Material (stems, leaves, etc to be processed)
10	Seed
11	Plant Tissue
12	Mature Plant
13	Flower Lot
14	Other Plant Material Lot
15	Bubble Hash
16	Hash
17	Hydrocarbon Wax
18	CO2 Hash Oil
19	Food Grade Solvent Extract
20	Infused Dairy Butter or Fat in Solid Form
21	Infused Cooking Oil
22	Solid Marijuana Infused Edible
23	Liquid Marijuana Infused Edible
24	Marijuana Extract for Inhalation

25	Marijuana Infused Topicals
26	Sample Jar
27	Waste
28	Usable Marijuana

Unique Identifiers

The system will generate unique identifiers for all plants and inventory. Plants will be assigned random sixteen digit identifiers. Inventory items (e.g. lots, batches, etc.) will also be provided identifiers, with the first nine digits representing the UBI number of the producer or processor that is creating the item.

Convenience Functions

A number of convenience functions have been removed to facilitate a quicker implementation timeline for third party integrators. A future specification may re-implement these to further improve data integrity.

Chapter 1: Authentication

In this chapter, you'll learn how to:

- ✓ Communicate with the traceability system
- ✓ Authenticate
- ✓ Create and modify users
- ✓ Elevate privileges, when necessary

Every request begins with with "json".. The current iteration of our API is now at 4.0. It is **strongly** recommended that every application specify this with every request. We do anticipate future changes and specifying the API will ensure your application does not receive errors when features are added or deprecated, but not entirely removed. Otherwise, the system will assume you are referencing the latest version. Every API request has an action associated with it. Any request that does not specify an action will automatically be rejected. Improperly formatted JSON requests will be rejected. When in doubt, see: http://jsonlint.com/. So, at bare minimum, a request should appear as follows:

```
{
  "JSON": {
    "API": "4.0",
    "action": "foo"
  }
}
```

The request should be sent as a raw POST request (URL to follow) of the type text/JSON. The result will also be of text/JSON type.

login

When registering with the WSLCB, an account administrator will receive a password in their email that will grant full access. This email address and password can then be shared, stored or utilized by a commercial application to initially authenticate with the traceability system.

Parameters:

action	variable length text field
username	variable length text field
password	variable length text field
license_number	variable length text field


```
{
 "JSON": {
  "API": "4.0",
  "action": "login",
  "password": "foobar",
  "license_number": "000000009",
  "username": "username@domain.com"
 }
}
```


A client should login with their username, password and the 9 digit UBI number of their account. A successful authentication will result in the following:


```
{
 "JSON": {
  "admin": "1",
  "sessionid":
"2f58596cad6db73d6cdd599b11cd169263a54cd37dc75ae0bfefe0cd9c9
c571c107059f23fe8cf7d4572f4878b9e1d9821e097e9348aa7b59a31180
ab8c9e6c8",
  "time": "1384323370",
  "success": "1"
 }
}
```


Returned Parameters:

admin	Boolean value
sessionid	sha512 hex encoded string

time	Unix 32-bit integer timestamp
success	Boolean value

The admin parameter will indicate that the authenticated user is an administrator capable of creating other users, setting permissions, etc. The sessionid parameter can be used for future requests under the user who originally authenticated for quicker requests.

If an application is not interested in maintaining sessions, they may also choose to simply include the aforementioned values with the nosession parameter. For example:

```
{
 "JSON": {
  "API": "4.0",
  "action": "test",
  "password": "foobar",
  "license_number": "000000009",
  "username": "username@domain.com",
  "nosession": "1"
 }
}
```

By setting the nosession parameter to 1, requests can be made without creating a stateful session, if necessary.

During the course of a normal session, a session's credentials can also be temporarily elevated for the duration of the action by passing the super_user and super_password parameters.

```
{
```

```
 "JSON": {
   "API": "4.0",
   "action": "admin_action_example",
   "sessionid":
"2f58596cad6db73d6cdd599b11cd169263a54cd37dc75ae0bfe
fe0cd9c9c571c107059f23fe8cf7d4572f4878b9e1d9821e097e9
348aa7b59a31180ab8c9e6c8",
   "super_password": "foobar",
   "super_user": "username@domain.com",
   "param": "foo"
 }
}
```

If a function call returns 0 value for success, it will also set an <error>explanation</error> for easier error handling. In addition, it will also carry an <errorcode>1234</errorcode> for reference. This document does not **currently** have a detailed list of error codes. That will be forthcoming in the final draft for ease of debugging efforts. For brevity, all code examples hereafter will omit the sessionid parameter; but it is assumed that either that or the proper nosession credentials are provided for **every** request.

The application interface also supports a testing interface. If a licensee wishes to practice or a commercial application wishes to test their integration capabilities a request may include the <training>1</training> node within a request. Users cannot be created, modified or removed in training mode. They are automatically transposed from the production environment. Every user automatically has full capabilities in training mode; that is, there are no ACL controls (as the data is not real). If a session is created in training mode, and an attempt is made to perform an action in production mode (or vice versa) an invalid session will be triggered as they operate completely separate from one another. It will be up to the application to save state as to which mode the connection was initiated with. As can be seen below, training mode is easy to trigger:

```
{
 "JSON": {
```

```
  "API": "4.0",
  "training": "1",
  "action": "login",
  "password": "foobar",
  "license_number": "123456789",
  "username": "username@domain.com"
 }
}
```

user_add

Users with administrative privileges can add other users via the user_add function. As demonstrated below, each function is discrete and robust ACLs can be utilized by an integrating party.

Parameters:

action	variable length text field
new_username	variable length text field
new_password	variable length text field
new_permissions	nested field that includes boolean values for each permission

```
{
 "JSON": {
  "API": "4.0",
  "action": "user_add",
  "new_admin": "1",
  "new_password": "foobar",
  "new_username": "user1@domain.com",
  "new_permissions": {
   "employee_add": "1",
   "employee_modify": "1",
```

"employee_remove": "1",
"vehicle_add": "1",
"vehicle_modify": "1",
"vehicle_remove": "1",
"plant_destroy_schedule": "1",
"plant_destroy": "1",
"plant_harvest_schedule": "1",
"plant_waste_weigh": "1",
"plant_harvest": "1",
"plant_new": "1",
"plant_convert_to_inventory": "1",
"plant_cure": "1",
"plant_move": "1",
"plant_yield_modify": "1",
"inventory_new": "1",
"inventory_transfer": "1",
"inventory_adjust": "1",
"inventory_destroy_schedule": "1
",
"inventory_convert": "1",
"inventory_sample": "1",

"inventory_manifest": "1",
"inventory_check": "1",
"inventory_destroy": "1",
"inventory_move": "1",
"inventory_transfer_schedule": "1",
"inventory_transfer_modify": "1",
"inventory_create_lot": "1",
"inventory_split": "1",
"user_add": "1",
"user_modify": "1",
"user_remove": "1",
"location_add": "1",
"location_modify": "1",
"location_remove": "1",
"plant_room_add": "1",
"plant_room_modify": "1",
"plant_room_remove": "1",
"inventory_room_add": "1",
"inventory_room_modify": "1",
"inventory_room_remove": "1",
"sale_dispense": "1",

```
    "sale_void": "1",
    "sale_modify": "1",
    "sale_refund": "1"
   }
  }
}
```

Each permission should either be 1 for true, 0 for false. Any nested parameter for the new_permissions parameter that are not included shall be assumed to be 0.

Returned Parameters:

success	Boolean value

user_modify

Users with administrative privileges can modify other users via the user_modify function.

Parameters:

action	variable length text field
new_username	variable length text field
new_password	variable length text field
new_permissions	nested field that includes boolean values for each permission

```
{
 "JSON": {
   "API": "4.0",
   "action": "user_modify",
   "new_admin": "1",
   "new_password": "foobar",
   "new_username": "user1@domain.com",
   "new_permissions": "
…
"
  }
```

```
}
```

Returned Parameters:

success	Boolean value

user_remove

Users with administrative privileges can remove other users via the user_remove function. Please note: The initial user that was created with the license cannot be removed.

Parameters:

action	variable length text field
new_username	variable length text field

```
{
 "JSON": {
  "API": "4.0",
  "action": "user_remove",
  "new_username": "user1@domain.com"
 }
}
```

Returned Parameters:

success	Boolean value

Chapter 2: Employees & Vehicles

In this chapter, you'll learn how to:

- ✓ Add, modify and remove employees
- ✓ Add, modify and remove vehicles

employee_add

Every organization will need to input basic information on their employees when providing samples or submitting transport manifests. Organizations will not be required to provide comprehensive employee lists, but, rather, on an as-needed basis for actions requiring an employee identification.

Parameters:

action	variable length text field
employee_name	variable length text field
employee_id	unique variable length text field
birth_month	two character integer
birth_day	two character integer
birth_year	four character integer
hire_month	two character integer
hire_day	two character integer
hire_year	four character integer

```
{
 "JSON": {
  "API": "4.0",
  "action": "employee_add",
  "employee_name": "Joe Employee",
  "employee_id": "12345",
  "birth_month": "01",
  "birth_day": "01",
  "birth_year": "1980",
  "hire_month": "01",
  "hire_day": "01",
  "hire_year": "2014"
 }
```

```
}
```

Returned Parameters:

success	Boolean value

employee_modify

This function should be used to update an existing employee.

Parameters:

action	variable length text field
employee_name	variable length text field
employee_id	unique variable length text field
birth_month	two character integer
birth_day	two character integer
birth_year	four character integer
hire_month	two character integer
hire_day	two character integer
hire_year	four character integer

```
{
 "JSON": {
  "API": "4.0",
  "action": "employee_modify",
  "employee_name": " Joe Employee",
  "employee_id": "12345",
  "birth_month": "01",
  "birth_day": "01",
  "birth_year": "1980",
  "hire_month": "01",
  "hire_day": "01",
  "hire_year": "2014"
 }
}
```

Returned Parameters:

success	Boolean value

employee_remove

This function should be used to remove an employee.

Parameters:

action	variable length text field
employee_id	unique variable length text field

```
{
 "JSON": {
  "API": "4.0",
  "action": "employee_remove",
  "employee_id": "12345"
 }
}
```

Returned Parameters:

success	Boolean value

vehicle_add

Every organization will need to input basic information on their vehicles when submitting transport manifests. This includes an integer id number that should be associated with the vehicle and the associated information for that vehicle, including: Color, make, model, plate and VIN.

Parameters:

action	variable length text field
vehicle_id	unique integer
color	variable length text field
make	variable length text field
model	variable length text field
plate	variable length text field
vin	variable length text field

```
{
 "JSON": {
  "API": "4.0",
  "action": "vehicle_add",
```

```
   "vehicle_id": "2",
   "color": "Red",
   "make": "Ford",
   "model": "Mustang",
   "plate": "ABC124",
   "vin": "123242365566"
  }
}
```


Returned Parameters:

success	Boolean value

vehicle_modify

This function should be used to update an existing vehicle.

Parameters:

action	variable length text field
vehicle_id	unique integer
color	variable length text field
make	variable length text field
model	variable length text field
plate	variable length text field
vin	variable length text field


```
{
 "JSON": {
   "API": "4.0",
   "action": "vehicle_modify",
   "vehicle_id": "2",
   "color": "Blue",
   "make": "Ford",
   "model": "Mustang",
   "plate": "ABC124",
   "vin": "123242365566"
  }
}
```

Returned Parameters:

success	Boolean value

vehicle_remove

This function should be used to remove an employee.

Parameters:

action	variable length text field
vehicle_id	unique integer

```
{
 "JSON": {
  "API": "4.0",
  "action": "vehicle_remove",
  "vehicle_id": "2"
 }
}
```

Returned Parameters:

success	Boolean value

Chapter 3: Rooms

In this chapter, you'll learn how to:

- ✓ Add, modify and remove plant rooms
- ✓ Add, modify and remove inventory rooms

plant_room_add

Plant rooms represent a way to logically segregate plants in a specific location. These can include actual rooms inside of indoor facility or fields in an outdoor facility.

Parameters:

action	variable length text field
name	variable length text field
location	license number of location value
id	integer value

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_room_add",
  "name": "Veg 1",
  "id": "1",
  "location": "12345"
 }
}
```

Returned Parameters:

success	Boolean value

plant_room_modify

Plant rooms can be renamed or re-activated with this function.

Parameters:

action	variable length text field
name	variable length text field

location	license number of location value
id	integer value

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_room_modify",
  "name": "Veg 2",
  "id": "1",
  "location": "12345"
 }
}
```

Returned Parameters:

success	Boolean value

plant_room_remove

Plant rooms can be removed with this function.

Parameters:

action	variable length text field
location	license number of location value
id	integer value

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_room_remove",
  "id": "1"
 }
}
```

Returned Parameters:

success	Boolean value

inventory_room_add

Inventory rooms represent a way to logically segregate inventory in a specific location. This can offer a real-time representation not only of the overall on-hand amount of a specific item but also the amount in a specific area of a facility. A room can be designated as a quarantine room with this function, as well. At least one quarantine room is required for segregating inventory before transportation. A room identifier must always be greater than zero. The room 0 is reserved as a general identifier for inventory that has not been assigned to a room.

Parameters:

action	variable length text field
name	variable length text field
location	license number of location value
id	integer value
quarantine	Boolean value

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_room_add",
  "name": "Veg 1",
  "id": "1",
  "quarantine": "0",
  "location": "12345"
 }
}
```

Returned Parameters:

success	Boolean value

inventory_room_modify

Inventory rooms can be renamed or re-activated with this function.

Parameters:

action	variable length text field
name	variable length text field
location	license number of location value
id	integer value
quarantine	Boolean value

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_room_modify",
  "name": "Veg 2",
  "id": "1",
  "quarantine": "0",
  "location": "12345"
 }
}
```

Returned Parameters:

success	Boolean value

inventory_room_remove

Inventory rooms can be removed with this function.

Parameters:

action	variable length text field
location	license number of location value
id	integer value

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_room_remove",
  "id": "1"
 }
}
```

Returned Parameters:

success	Boolean value

Chapter 4: Plants

In this chapter, you'll learn how to:

- ✓ Add and remove plants
- ✓ Harvest and cure plants
- ✓ …and much, much more!

plant_new

The plant_new function will allow a cultivator to enter new plants into the traceability system. This function will require the strain, quantity, location, new room, whether the plant will be used as a mother plant (this can be toggled later if necessary) and the source identification number. The source identification number can be from one of the following inventory types: Clone, Seed, Mature Plant and Plant Tissue. Clone, Seed and Mature Plant are depletable inventory items in that any plant creation will automatically deduct from the count in inventory (so ensure that the quantity of new plants does not exceed that available from inventory).

Parameters:

action	variable length text field
strain	variable length text field
location	license number of location
room	integer value
source	text field representing unique identifier
quantity	integer value

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_new",
  "location": "12345",
  "source": "2288954595338316",
  "quantity": "2",
  "room": "1",
  "strain": "Blueberry"
 }
}
```

Return example:


```
{
 "JSON": {
  "barcode_id": [
   "6853296789574115",
   "6853296789574116"
  ],
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3278"
 }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
barcode_id	Array of 1 or more text fields representing the new unique identifiers attached to the plants

Transaction IDs are generated for every action which involves the submission of licensee data. These TIDs are used for audit purposes and should be maintained.

plant_move

The plant_move function will allow a cultivator to move plants from their current room to a new one.

Parameters:

action	variable length text field
room	integer value
barcodeid	Array of 1 or more text fields representing the plants to move


```
{
 "JSON": {
  "API": "4.0",
```

```
  "action": "plant_move",
  "barcodeid": [
   "6853296789574115",
   "6853296789574116"
  ],
  "room": "2"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value

plant_destroy_schedule

The plant_destroy_schedule function will allow a licensee to schedule for destruction a plant or set of plants. This event will begin a 72-hour waiting period before a plant_destroy function may be called on the plant(s).

Parameters:

action	variable length text field
reason	variable length text field
barcodeid	Array of 1 or more text fields representing the plants

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_destroy_schedule",
  "barcodeid": [
   "6853296789574115",
   "6853296789574116"
  ],
  "reason": "Mold"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

plant_destroy

The plant_destroy function will allow a licensee to destroy a plant or set of plants. Plants may only be destroyed after the waiting period has expired.

Parameters:

action	variable length text field
barcodeid	Array of 1 or more text fields representing the plants

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_destroy",
  "barcodeid": [
   "6853296789574115",
   "6853296789574116"
  ]
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

plant_harvest_schedule

The plant_harvest_schedule function will notify the traceability system of intent to begin harvesting a plant or set of plants. This notification must occur before the plant_harvest is called on these plants.

Parameters:

action	variable length text field

barcodeid	Array of 1 or more text fields representing the plants

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_harvest_schedule",
  "barcodeid": [
   "6853296789574115",
   "6853296789574116"
  ]
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

plant_harvest

The plant_harvest function will begin the process of harvesting a plant. This will move said plant from the "growing" phase to the "drying" phase. During this process, a cultivator must take, at a minimum, a wet weight of the plant. In addition, a cultivator may also gather two additional derivatives defined by their inventory type. Specifically, the system requires inventory type 6 (Flower) and optionally allows type 9 (Other Plant Material) and type 27 (Waste).

Harvests can be partial, as well. In other words, if part of the plant is harvested and the rest of the plant will be processed later (commonly known as re-flowering), then the collectadditional parameter should be 1. This will inform the traceability system to expect another additional wet weight.

Each harvest event should be on a per-plant basis. So every individual plant will need its own wet weight reported. Both Other Plant Material and Waste collected during this process will receive random unique identifiers. For Other Plant Material, this will facilitate the process of creating a lot. For Waste, this will allow a user to accumulate waste in a traceable manner and schedule a destruction event at a later point.

Parameters:

action	variable length text field
collectiontime	Optional, Unix 32-bit integer timestamp, defaults to current time
barcodeid	unique identifier of the plant
weights	Array of 1 or more nodes containing weight information
amount	decimal value
invtype	integer value representing the derivative type
uom	variable length text field. Valid values are: g, mg, kg, oz, lb. These represent: grams, milligrams, kilograms, ounces and pounds.
collectadditional	Keeps the plant in the growing phase and allows the user to take another wet weight of the plant(s) at a later point that will compound to the original wet weight.
new_room	Optional, will move the now drying plant(s) to another plant room.
room	integer, room the collection occurred in
location	license number of location

Example:


```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_harvest",
  "barcodeid": "9318094993507695",
  "collectadditional": "0",
  "location": "12345",
  "room": "2",
  "new_room": "3",
  "weights": [
   {
```

```
      "amount": "250.00",
      "invtype": "6",
      "uom": "g"
     },
     {
      "amount": "500.00",
      "invtype": "9",
      "uom": "g"
     },
     {
      "amount": "125.00",
      "invtype": "27",
      "uom": "g"
     }
   ]
  }
}
```


Returns:


```
{
 "JSON": {
  "derivatives": [
   {
    "barcode_id": "0358560579655604",
    "barcode_type": "9"
   },
   {
    "barcode_id": "0358560579655605",
    "barcode_type": "27"
   }
  ],
  "sessiontime": "1384487873",
  "success": "1",
  "transactionid": "3284"
 }
```

}

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
derivatives	Array of 1 or more nodes containing new identifiers with their associated inventory types.
barcode_id	New identifier for the inventory specified by barcode_type.
barcode_type	Specifies the type of derivative.

plant_waste_weigh

The plant_waste_weigh function will allow a cultivator to take a general waste weight for destruction accountability at a later point. General leaf, stem, veg trimming, etc. collection can thus be facilitated in a more generalized fashion without unduly burdening a licensee.

The return inventory will be typed as 27 and must be scheduled for destruction at a later point.

Parameters:

action	variable length text field
collectiontime	Optional, Unix 32-bit integer timestamp, defaults to current time
weight	decimal value
uom	variable length text field. Valid values are: g, mg, kg, oz, lb. These represent: grams, milligrams, kilograms, ounces and pounds.
location	license number of location

Example:

```
{
 "JSON": {
  "API": "4.0",
```

```
  "action": "plant_waste_weigh",
  "location": "12345",
  "weight": "250.00",
  "uom": "g"
 }
}
```


Returns:


```
{
 "JSON": {
  "barcode_id": "0358560579655604",
  "barcode_type": "27",
  "sessiontime": "1384487873",
  "success": "1",
  "transactionid": "3286"
 }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
barcode_id	New identifier for the inventory specified by barcode_type.
barcode_type	Specifies the type of derivative, always 27.

plant_cure

The plant_cure function will begin the process of curing a plant. This will move said plant from the drying phase to inventory. During this process, a cultivator must take, at a minimum, a dry weight of the plant. In addition, a cultivator may also gather additional derivatives defined by their inventory type. Specifically, the system requires inventory type 6 (Flower) and optionally allows type 9 (Other Plant Material) and type 27 (Waste).

If the cultivator is doing a partial harvest/cure, the plant can pass through this function again to accumulate an additional dry weight. If the cultivator is re-flowering, ensure the collectadditional field is set to 1.

Parameters:	
action	variable length text field
collectiontime	Optional, Unix 32-bit integer timestamp, defaults to current time
barcodeid	unique identifier of the plant
weights	Array of 1 or more nodes containing weight information
amount	decimal value
invtype	integer value representing the derivative type
uom	variable length text field. Valid values are: g, mg, kg, oz, lb. These represent: grams, milligrams, kilograms, ounces and pounds.
collectadditional	Keeps the plant in the growing phase and allows the user to take another wet weight of the plant(s) at a later point that will compound to the original wet weight.
room	integer, room the collection occurred in
location	license number of location

Example:


```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_cure",
  "barcodeid": "9992776458335982",
  "collectadditional": "0",
  "location": "12345",
  "room": "2",
  "weights": [
   {
    "amount": "250.00",
    "invtype": "6",
    "uom": "g"
```

```
      },
      {
       "amount": "500.00",
       "invtype": "9",
       "uom": "g"
      },
      {
       "amount": "125.00",
       "collected": "0",
       "invtype": "27",
       "uom": "g"
      }
    ]
  }
}
```


Returns:


```
{
 "JSON": {
  "derivatives": [
    {
     "barcode_id": "0358560579655604",
     "barcode_type": "6"
    },
    {
     "barcode_id": "0358560579655605",
     "barcode_type": "9"
    }
  ],
  "sessiontime": "1384487873",
  "success": "1",
  "transactionid": "3290"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
derivatives	Array of 1 or more nodes containing new identifiers with their associated inventory types.
barcode_id	New identifier for the inventory specified by barcode_type.
barcode_type	Specifies the type of derivative.

plant_convert_to_inventory

The plant_convert_to_inventory function will allow a licensee to convert a plant that is growing (but not flowering) into an inventory item that can then be transferred and sold. Once converted, the new item will keep its identifier but will now have an inventory type of 12 (Mature Plant).

Parameters:

action	variable length text field
barcodeid	Array of 1 or more text fields representing the plants to convert

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_convert_to_inventory",
  "barcodeid": [
    "6853296789574125",
    "6853296789574126"
  ]
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

plant_yield_modify

The plant_yield_modify function will allow direct access to modify previously stored values for harvest and cure collections. The user will need to specify one transaction at a time. The integrator is, of course, free to hide this from the end-user with multiple API calls behind the scenes if they display the capability to modify collected values in a unique or innovative way.

The user can, however, specify all values that would have been specifiable at the time of the original transaction. That is, if the transaction relates to the plant_harvest, wet weight and any derivative can be specified. If the original transaction was a plant_cure, dry weight could be specified, instead. Only values that are included will be modified. If a user wishes to zero out a value, it must be declared. Null or absent values will retain their previous values.

Parameters:

action	variable length text field
collectiontime	Optional, Unix 32-bit integer timestamp, defaults to current time
transactionid	integer, the transaction to correct
weights	Array of 1 or more nodes containing weight information
amount	Optional, decimal value
invtype	integer value representing the derivative type
uom	variable length text field. Valid values are: g, mg, kg, oz, lb. These represent: grams, milligrams, kilograms, ounces and pounds.

Example:

```
{
 "JSON": {
  "API": "4.0",
  "action": "plant_yield_modify",
  "transactionid": "3290",
  "weights": {
   "amount": "450.00",
   "invtype": "6",
   "uom": "g"
```

```
    }
  }
}
```


Returns:


```
{
  "JSON": {
    "sessiontime": "1384487873",
    "success": "1",
    "transactionid": "3309"
  }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

Chapter 5: Inventory

In this chapter, you'll learn how to:

- ✓ Adjust and audit inventory
- ✓ Create new inventory
- ✓ Convert inventory
- ✓ Perform inventory lookups

inventory_adjust

The inventory_adjust function will allow a licensee to adjust the amount or quantity of an inventory item. The type field can represent one of the following: 1 (General Inventory Audit), 2 (Theft), 3, (Seizure by Federal, State, Local or Tribal Law Enforcement), 4 (Correcting a mistake)

Parameters:

action	variable length text field
barcodeid	inventory identifier
quantity	integer value, new quantity
uom	variable length text field. Valid values are: g, mg, kg, oz, lb, each. These represent: grams, milligrams, kilograms, ounces, pounds, each. If weighable, grams are assumed if omitted. If non-weighable, each is assumed.
reason	variable length text field explaining in greater detail the reason for the removal or addition of inventory
type	Integer value representing the type of adjustment.

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_adjust",
  "barcodeid": "6647455983218747",
  "quantity": "690",
  "reason": "Testing",
  "type": "1"
 }
}
```


Return example:


```
{
 "JSON": {
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3311"
 }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

inventory_destroy_schedule

The inventory_destroy_schedule function will notify the traceability system of intent to destroy an inventory item. Per current rules, this function can only (currently) be called by producers and processors.

Parameters:

action	variable length text field
barcodeid	Array of 1 or more text fields representing the plants
reason	reason for the destruction

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_destroy_schedule",
  "barcodeid": [
   "6853296789574115",
   "6853296789574116"
  ],
  "reason": "Mold"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

inventory_destroy

The inventory_destroy function will allow a licensee to destroy an item that has been previously scheduled for destruction.

Parameters:

action	variable length text field
barcodeid	inventory identifier
reason	reason for the removal or addition of inventory
health	Boolean value, indicates if the removal is due to health concerns

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_destroy",
  "barcodeid": "6647455983218747"
 }
}
```

Return example:


```
{
 "JSON": {
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3411"
 }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

inventory_move

The inventory_move function will update the current room for the specified inventory items. Essentially, it allows a user to move inventory from one room to another.

Parameters:

action	variable length text field
data	Array of 1 or more nodes containing inventory information
barcodeid	inventory identifier
room	Integer value, represents the identification number of a room


```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_move",
  "data": [
   {
    "barcodeid": "7480211204033809",
    "room": "1"
   },
   {
    "barcodeid": "7480211204033808",
```

```
      "room": "1"
    }
  ]
 }
}
```

Return example:

```
{
 "JSON": {
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3626"
 }
}
```

inventory_check

The inventory_check function can be used to perform a cursory lookup on an item before an inbound inventory_transfer from an outside licensee. It will pull various pieces of inventory on the inventory identifiers specified in the request. This information can include: strain, quantity available, usable weight (if applicable), product (if applicable) and inventory type.

Parameters:

action	variable length text field
barcodeid	Array of 1 or more text fields representing the inventory to lookup

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_check",
  "barcodeid": [
   "6853296789574115",
   "6853296789574116"
  ]
 }
```

}

Returned Parameters:

success	Boolean value
data	Array of 1 or more nodes containing inventory information
barcode_id	inventory identifier
strain	variable length text field
product	variable length text field
quantity	decimal value
usableweight	decimal value (in grams).
inventorytype	integer value based on pre-defined inventory types

Return example:


```
{
 "JSON": {
  "data": {
   "barcode_id": "8919990967962719",
   "invtype": "28",
   "quantity": "10",
   "usableweight": "3.50",
   "strain": "Blueberry"
  },
  "success": "1"
 }
}
```


inventory_new

The inventory_new function can be used to create new inventory not previously entered into the system. This function is ONLY accessible to a licensee that has been designated as a producer may only be used for the first 15 days of operation. Subsequent calls to this function will be denied. In addition, only four types may be provided to this function: Seed, Clone, Mature Plant and Plant Tissue.

Parameters:

action	variable length text field
location	license number of location
data	Array of 1 or more nodes containing new inventory information
strain	variable length text field
quantity	integer value
invtype	integer, corresponds to the inventory type system

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_new",
  "data": {
   "invtype": "12",
   "quantity": "50",
   "strain": "Blueberry"
  },
  "location": "12345"
 }
}
```

Return example:

```
{
 "JSON": {
  "barcode_id": [
   "6853296789574115",
   "6853296789574116"
  ],
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3278"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
barcode_id	Array of 1 or more text fields representing the new unique identifiers attached to the inventory items

inventory_manifest

The inventory_manifest function will notify the traceability system of intent to transfer an inventory item. This function will need to be called in instances of transfers from one licensee to another. It will also need to be called for licensees which possess multiples licenses (e.g. Producer + Processor) that possess different license numbers. For internal transfers (e.g. from one part of a facility to another), there is no need to quarantine and schedule a transfer.

Parameters:

action	variable length text field
barcodeid	Array of 1 or more text fields representing the items to be transferred
employee_id	variable length text field
vehicle_id	integer value
approximate_departure	Unix 32-bit integer timestamp, approximate departure time
approximate_arrival	Unix 32-bit integer timestamp, approximate arrival time
approximate_route	variable length text field, route that will be used
vendor_license	license number of vendor the item(s) are being transferred to
new_room	Optional, can specify the item(s) have been placed into e.g. a quarantine room.

Example:

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_manifest",
```

```
    "barcodeid": [
      "6853296789574115",
      "6853296789574116"
    ],
    "employee_id": "23468",
    "vehicle_id": "2",
    "approximate_departure": "1384476925",
    "approximate_arrival": "1384486925",
    "approximate_route": "Turn left on Main St.",
    "vendor_license": "25678787644"
  }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
barcode_id	Unique identifier attached to the manifest

inventory_transfer

The inventory_transfer function can be used to transfer inventory that already exists in the system. A manifest must be filed prior to transfer if being transferred to a license number other than the one that currently possesses the item.

Parameters:

action	variable length text field
vendor_license	variable length text field
data	Array of 1 or more nodes containing inventory information
barcodeid	inventory identifier
price	Optional if inter-UBI transfer, decimal value that indicates how much the item was sold for INCLUDING excise tax.

```
{
  "JSON": {
```

```
  "API": "4.0",
  "action": "inventory_transfer",
  "data": {
   "barcodeid": "6853296789574115",
   "price": "100.00"
  }
 }
}
```


Return example:


```
{
 "JSON": {
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3778"
 }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

inventory_transfer_modify

The inventory_transfer_modify function will allow a user to modify the price recorded for an inventory transfer sale. This can be used before filing a monthly report if a line item mistake is noticed and needs to be corrected.

Parameters:

action	variable length text field
transactionid	integer value
barcodeid	inventory identifier
price	Decimal value representing the price paid INCLUDING the excise tax but NOT including any other taxes that may be applicable.

item_number	Optional, integer, should be provided if multiple line items of the same barcode were included in one sale. 0 would represent the first item (in the order submitted to the system), 1 the next, etc.

Example:

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_transfer_modify",
  "transactionid": "3590",
  "barcodeid": "6647455983218749",
  "price": "15.00"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

inventory_create_lot

The inventory_create_lot function will allow a user to combine inventory types 6 (Flower) and 9 (Other Plant Material) into lots as mandated by rules. The return types will be 13 (Flower Lot) and 14 (Other Plant Material Lot), respectively.

Parameters:

action	variable length text field
strain	variable length text field
lot_quantity	decimal value, new quantity of combined items
lot_quantity_uom	variable length text field. Valid values are: g, mg, kg, oz, lb, each. These represent: grams, milligrams, kilograms, ounces, pounds, each.

data	Array of 1 or more nodes containing inventory information
barcodeid	inventory identifier
remove_quantity	integer value, quantity to remove. Does not need to be remaining quantity (can be a partial combination).
remove_quantity_uom	variable length text field. Valid values are: g, mg, kg, oz, lb, each. These represent: grams, milligrams, kilograms, ounces, pounds, each.


```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_create_lot",
  "lot_quantity": "945",
  "data": [
    {
     "barcodeid": "6647455983218747",
     "remove_quantity": "693.00"
    },
    {
     "barcodeid": "5723224643296982",
     "remove_quantity": "252.00"
    }
  ],
  "strain": "Blueberry"
 }
}
```


Return example:


```
{
 "JSON": {
  "sessiontime": "1384476925",
  "barcode_id": "5723224643296983",
```

```
   "barcode_type": "13",
   "success": "1",
   "transactionid": "3312"
  }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
barcode_id	text field representing new unique identifier
barcode_type	integer representing new lot type

inventory_split

The inventory_split function will allow a user to split inventory items into sub lots or sub batches. For example, if a user has a lot of Flower and only wishes to sell half of it, they would need to first create a sub lot using this function. Then, with the new lot number, they can sell the desired amount. Multiple lots or batches can be specified at a time, however, keep in mind they will not be combined. Rather, each one will receive a new sub-lot or sub-batch number.

Parameters:

action	variable length text field
data	Array of 1 or more nodes containing inventory information
barcodeid	inventory identifier
remove_quantity	integer value, quantity to remove. Does not need to be remaining quantity (can be a partial combination).
remove_quantity_uom	variable length text field. Valid values are: g, mg, kg, oz, lb, each. These represent: grams, milligrams, kilograms, ounces, pounds, each.


```
{
 "JSON": {
  "API": "4.0",
```

```
   "action": "inventory_split",
   "data": [
     {
      "barcodeid": "6647455983218747",
      "remove_quantity": "693.00"
     },
     {
      "barcodeid": "5723224643296982",
      "remove_quantity": "252.00"
     }
   ]
  }
}
```

Return example:

```
{
 "JSON": {
   "sessiontime": "1384476925",
   "barcode_id": [
     "5723224643296983",
     "5723224643296984"
   ],
   "success": "1",
   "transactionid": "3312"
  }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
barcode_id	text fields representing new unique identifier, returned in the order of the input identifiers

inventory_convert

The inventory_convert function will allow a user to convert one type of item to another. The system allows for multiple sources. So, for example, a processor may use part of various Other Plant Material Lots in producing a batch of hash oil. Certain derivatives may not be strain specific, so entering a strain is optional under those circumstances. Product name is optional when it is not the end product. If the derivative item will be sold to a consumer (that is, inventory types 22,23,24,25) and is not regular usable marijuana (type 28), then a product will be required (e.g. Cookie, Brownie, etc).

Parameters:

action	variable length text field
data	Array of 1 or more nodes containing inventory information
barcodeid	inventory identifier
remove_quantity	integer value, quantity to remove. Does not need to be remaining quantity (can be a partial combination).
remove_quantity_uom	variable length text field. Valid values are: g, mg, kg, oz, lb, each. These represent: grams, milligrams, kilograms, ounces, pounds, each.
waste	decimal value, amount of waste produced by the process, if any
waste_uom	Valid values are: g, mg, kg, oz, lb. These represent: grams, milligrams, kilograms, ounces, pounds.
derivative_type	Inventory type of derivative item
derivative_quantity	decimal value, quantity of new derivative after conversion
derivative_quantity_uom	Valid values are: g, mg, kg, oz, lb, each. These represent: grams, milligrams, kilograms, ounces, pounds, each.
derivative_usable	decimal value, quantity of usable marijuana in new product after conversion
derivative_usable_uom	Valid values are: g, mg, kg, oz, lb, each. These represent: grams,

	milligrams, kilograms, ounces, pounds, each.
derivative_strain	Optional, variable length text field
derivative_product	Optional, variable length text field

Example:


```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_convert",
  "data": {
   "barcodeid": "6647455983218747",
   "remove_quantity": "25.00"
  },
  "waste": "15.00",
  "derivative_quantity": "10.00",
  "derivative_inventory_type": "18"
 }
}
```


Return example:


```
{
 "JSON": {
  "derivatives": [
   {
    "barcode_id": "0358560579655606",
    "barcode_type": "18"
   },
   {
    "barcode_id": "0358560579655605",
    "barcode_type": "27"
   }
  ]
 }
```

}

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp
derivatives	Array of 1 or more nodes containing new identifiers with their associated inventory types.
barcode_id	New identifier for the inventory specified by barcode_type.
barcode_type	Specifies the type of derivative.

inventory_sample

The inventory_sample function will allow a user to provide samples as allowed by law. Specifically, samples can be provided to employees for quality assurance purposes or to vendors for the purposes of negotiating a sale. Either employee_id or vendor_license should be provided; but not both.

Parameters:

action	variable length text field
barcodeid	inventory identifier
employee_id	Optional, variable length text field
vendor_license	Optional, variable length text field representing license number of receiving entity
quantity	decimal value, quantity of old product before conversion
quantity_uom	Valid values are: g, mg, kg, oz, lb, each. These represent: grams, milligrams, kilograms, ounces, pounds, each.

Example:

```
{
 "JSON": {
  "API": "4.0",
  "action": "inventory_sample",
```

```
    "barcodeid": "6647455983218747",
    "quantity": "1.00",
    "employee_id": "12356"
  }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

Chapter 6: Sales

In this chapter, you'll learn how to:

- ✓ Deduct inventory for a sale
- ✓ Void a sale
- ✓ Refund a sale

sale_dispense

The sale_dispense function will allow a user to deduct items from inventory through the sales process. Since all items sold must be pre-packaged, units will be assumed to be "each".

Parameters:	
action	variable length text field
data	Array of 1 or more nodes containing inventory information
barcodeid	inventory identifier
quantity	integer value, quantity to remove
price	Decimal value representing the price paid INCLUDING the excise tax but NOT including any other taxes that may be applicable.

Example:


```
{
 "JSON": {
  "API": "4.0",
  "action": "sale_dispense",
  "data": [
    {
     "barcodeid": "6647455983218747",
     "quantity": "1.00",
     "price": "5.00"
    },
    {
     "barcodeid": "6647455983218749",
     "quantity": "1.00",
     "price": "15.00"
    }
  ]
 }
}
```


Return example:


```
{
 "JSON": {
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3312"
 }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

sale_void

The sale_void function will reverse items that have been sold to a customer and return the items to inventory. A refund should be used, instead, when the return is not being used to simply fix a mistake.

Parameters:

action	variable length text field
transactionid	integer value

Example:

```
{
 "JSON": {
  "API": "4.0",
  "action": "sale_void",
  "transactionid": "3590"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

sale_modify

The sale_modify function will allow a user to modify the price recorded for a sale. This can be used before filing a monthly report if a line item mistake is noticed and needs to be corrected.

Parameters:

action	variable length text field
transactionid	integer value
barcodeid	inventory identifier
price	Decimal value representing the price paid INCLUDING the excise tax but NOT including any other taxes that may be applicable.
item_number	Optional, integer, should be provided if multiple line items of the same

	barcode were included in one sale. 0 would represent the first item (in the order submitted to the system), 1 the next, etc.

Example:

```
{
 "JSON": {
  "API": "4.0",
  "action": "sale_modify",
  "transactionid": "3590",
  "barcodeid": "6647455983218749",
  "price": "15.00"
 }
}
```

Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

sale_refund

The sale_refund function is nearly identical to sale_dispense except that it for items to selectively come back into inventory from a sale. This can take place at any time period after the original sale and will reflect on current sales as opposed to affecting previously reported data. You must specify both a transactionid and one or more identifiers. Retailers are not currently allowed by rule to destroy product, so if an open item is received it must be scheduled for transfer back to the processor for destruction.

Parameters:

action	variable length text field
data	Array of 1 or more nodes containing inventory information
barcodeid	inventory identifier
quantity	integer value, quantity to bring in.
price	Negative decimal value representing the price paid INCLUDING the

excise tax but NOT including any other taxes that may be applicable.

Example:


```
{
 "JSON": {
  "API": "4.0",
  "action": "sale_refund",
  "data": [
   {
    "barcodeid": "6647455983218747",
    "quantity": "1.00",
    "price": "-5.00"
   },
   {
    "barcodeid": "6647455983218749",
    "quantity": "1.00",
    "price": "-15.00"
   }
  ]
 }
}
```


Return example:


```
{
 "JSON": {
  "sessiontime": "1384476925",
  "success": "1",
  "transactionid": "3312"
 }
}
```


Returned Parameters:

success	Boolean value
transactionid	integer value
sessiontime	Unix 32-bit integer timestamp

Chapter 7: Testing

In this chapter, you'll learn how to:

- ✓ Send lab results directly from a laboratory

Reserved

Chapter 8: Synchronization

In this chapter, you'll learn how to:

- ✓ Download current plants, inventory, etc. stored in traceability system
- ✓ Receive notifications of inventory seizures, etc.
- ✓ Assist a licensee transition from the state interface to a commercial application

Reserved